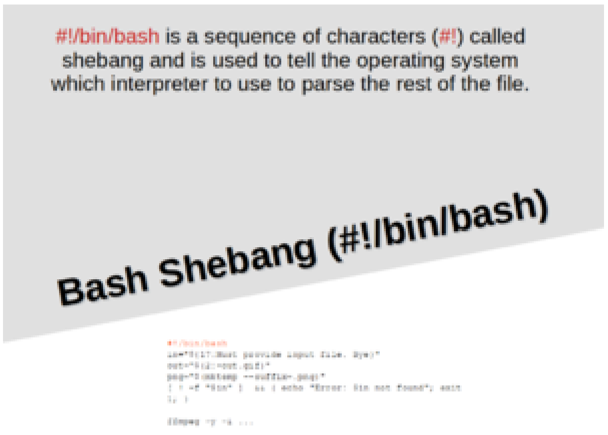
Shebang

← Hello, World! Tutorial • **Home** • Shell Comments →

You will always see `#!/bin/bash` or `#!/usr/bin/env bash` as the first line when writing or reading bash scripts. Shebang starts with **#!** characters and the path to the bash or other interpreter of your choice. Let us see what is Shebang in Linux and Unix bash shell scripts.


#!/bin/bash is a sequence of characters (#!) called shebang and is used to tell the operating system which interpreter to use to parse the rest of the file.
Bash Shebang (#!/bin/bash)

#!/bin/bash is a sequence of characters (#!) called shebang and is used to tell the Linux OS which interpreter to use to parse the rest of the file.

Contents

What is bash shebang?

The **#!** syntax is used in scripts to indicate an interpreter for execution under UNIX / Linux operating systems. The directive must be the first line in the Linux shell script and must start with shebang `#!`. You can add argument after the shebang characters, which is optional. Make sure the interpreter is the full path to a binary file. For example: /bin/bash.

Syntax

The syntax is:

```
#!/path/to/interpreter [arguments]
#!/path/to/interpreter -arg1 -arg2
```

Examples

```
#!/usr/bin/env bash
```

Perl and python examples:

```
#!/usr/bin/perl
```

OR

```
#!/usr/bin/python
```

OR

```
#!/usr/bin/python3
```

Starting a Script With #!

1. It is called a shebang or a "bang" line.
2. It is nothing but the absolute path to the Bash interpreter.
3. It consists of a number sign and an exclamation point character (#!), followed by the full path to the interpreter such as /bin/bash.
4. All scripts under Linux execute using the interpreter specified on a first line[1].
5. Almost all bash scripts often begin with #!/bin/bash (assuming that Bash has been installed in /bin)
6. This ensures that Bash will be used to interpret the script, even if it is executed under another shell[2].
7. The shebang was introduced by Dennis Ritchie between Version 7 Unix and 8 at Bell Laboratories. It was then also added to the BSD line at Berkeley [3].

Benefits [4].

1. It makes shell scripts more like actual executable files,
2. because they can be the subject of 'exec.'
3. If you do a 'ps' while such a command is running, the real name appears instead of 'sh' or 'bash'. Likewise, system accounting is done based on the real name.
4. Shell scripts can be set-user-ID.
5. It is simpler to have alternate shells available; e.g. if you like the Berkeley csh, there is no question about which shell is to interpret a file.
6. It will allow other interpreters to fit in more smoothly.

```
bash your-script.ksh
## or ##
bash /path/to/your-script.ksh
```

Although you can override and ignore the default Shebang set in the script, it may produce unexpected behaviour when overriding the shell interpreter. So be careful with overriding the shebang line.

/bin/sh

For a system boot script, use /bin/sh:

```
#!/bin/sh
```

sh is the standard command interpreter for the system. The current version of sh is in the process of being changed to conform with the POSIX 1003.2 and 1003.2a specifications for the shell.

An example of /bin/sh script

- /etc/init.d/policykit

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          policykit
# Required-Start:    $local_fs
# Required-Stop:     $local_fs
# Default-Start:     2 3 4 5
# Default-Stop:
# Short-Description: Create PolicyKit runtime directories
# Description:       Create directories which PolicyKit needs at runtime,
#                    such as /var/run/PolicyKit
### END INIT INFO

# Author: Martin Pitt <martin.pitt@ubuntu.com>

case "$1" in
  start)
        mkdir -p /var/run/PolicyKit
        chown root:polkituser /var/run/PolicyKit
        chmod 770 /var/run/PolicyKit
    ;;
  stop|restart|force-reload)
    ;;
  *)
    echo "Usage: $SCRIPTNAME {start|stop|restart|force-reload}" >&2
    exit 3
    ;;
esac

:
```

running user's $PATH variable.

```
#!/usr/bin/env bash
# Purpose: Mount glusterfs at boot time
#          Must run as root
# Author: Vivek Gite
# --------------------------------------
p='gfs01:/gvol01'

mount | grep -wq "^${p}"

if [ $? -ne 0 ]
then
    mount -t glusterfs "$p" /sharedwww/
fi
```

Summing up

I hope you know what Shebang is and how to use it in your Bash scripts under Linux and Unix-like systems with these examples. Do check the following resources:

See also

- Explain: #!/bin/bash (https://www.cyberciti.biz/faq/binbash-interpreter-spoofing/) - or #!/bin/bash -- In A Shell Script
- Make Linux/Unix Script Portable With #!/usr/bin/env As a Shebang (https://www.cyberciti.biz/tips/finding-bash-perl-python-portably-using-env.html)

References

1. Howto Make Script More Portable With #!/usr/bin/env As a Shebang (https://www.cyberciti.biz/tips/finding-bash-perl-python-portably-using-env.html) FAQ by nixCraft.
2. Bash man page and the official documentation.
3. extracts from 4.0BSD (http://www.in-ulm.de/~mascheck/various/shebang/sys1.c.html) /usr/src/sys/newsys/sys1.c.
4. extracts from 4.0BSD (http://www.in-ulm.de/~mascheck/various/shebang/sys1.c.html) /usr/src/sys/newsys/sys1.c.

← Hello, World! Tutorial • **Home** • Shell Comments →